Associating the Randomized Bluetooth MAC Addresses of a Device

Loïc Jouans, Aline Carneiro Viana, Nadjib Achir, Anne Fladenmuller

Associating the Randomized Bluetooth MAC Addresses of a Device

Loïc Jouans*†§, Aline Carneiro Viana†, Nadjib Achir†‡, Anne Fladenmuller§

* *ENS de Lyon*
Lyon, France
loic.jouans@ens-lyon.org

† *Inria*
Palaisau, France
{aline.viana, nadjib.achir}@inria.fr

‡ *Université Sorbonne Paris Nord*
Paris, France
nadjib.achir@univ-paris13.fr

§ *Sorbonne Université*
Paris, France
anne.fladenmuller@lip6.fr

***Abstract*—Bluetooth devices naturally emit many public signals. It opens new paths for passive mobility analysis, and allows building larger datasets direly needed by the research on mobile systems, but also raises new practical challenges. One of them is the correlation between the public packets and the emitters. The Bluetooth standard forces devices to regularly change the identifier they carry in the public packets, but we show in this paper it does not prevent from correlating packets from a given sender. The number of devices changing their MAC address at any given time is expectedly small; we only need to find a property differentiating a handful device at a time for MAC association. In this paper, we propose such a property and demonstrate the efficiency of the association strategy on Bluetooth Low Energy.**

***Index Terms*—Bluetooth Low Energy, Public packet correlation, Passive sensing**

I. Introduction

Research on mobile systems involves the analysis of spatiotemporal data. Unfortunately, gathering data is hard: recruiting volunteers demands considerable technical, logistic, and administrative efforts, and the subjects are generally an inhomogeneous sample of the population. All these drawbacks are inherent to active, intrusive data collection methods.

Against these issues, passive measurement strategies are a good contender, limiting the selection bias and expanding the potential target pool by orders of magnitude. They all draw upon the same principle: rebuilding information from the signals connected devices naturally emit. In this context, we could infer the mobility of the devices by leveraging their Wi-Fi, cellular, or Bluetooth signals. But cellular signals are fully ciphered and Wi-Fi signals are not regular enough for precise trajectory reconstruction; in contrast, Bluetooth (public) packets are regular, not ciphered, and practically every connected device emit them. Bluetooth, and more precisely its Low Energy variant, is a good candidate.

The price for these advances is a set of new challenges, one of the first being the device anonymization methods that modern Bluetooth protocols use. Most devices run the Bluetooth Low Energy (BLE) variant. With BLE turned on, a device shows it exists by sending public *advertising packets*, which contain an IEEE 802 MAC address [1][2, p. 69]. To protect the user and avoid displaying a unique identifier, the BLE protocols force devices to regularly change their public MAC address: it is the *MAC randomization process*. Being randomized, the MAC addresses are not private data, while regular MAC addresses may be, according to GDPR[1]. From BLE 4 to BLE 5.1, the protocols require devices to keep their MAC addresses for at least 15 minutes [3, p. 1756][4, p 2226].

This paper focuses on the MAC randomization process. It is a widely supported feature, independently of the superjacent OS. Nevertheless, current works in the literature either focus on devices not scrupulously respecting the standard [5], [6], or on the weaknesses the OS provider involuntarily introduces [7], [8]. Such strategies, though dealing with the MAC randomization process, target specific types of devices or OS versions; they reduce the extent of the device identification and the size of a mobility dataset. Instead, this paper targets any Bluetooth signal devices publicly emit. *To the best of our knowledge, we are the first to bring a solution to associate Bluetooth randomized MAC addresses together, denoted as an* ***association strategy.*** Such a strategy will allow the scientific community to study device mobility despite the Bluetooth MAC randomization process.

We introduce an association strategy that leverages the long timespan between MAC changes. *Over a small population, the number of devices that will change their MAC address at the same time is expectedly small. To differentiate these devices, we only need a property that can discriminate a handful of devices at a time*, in opposition to the previously cited papers that search for a device specific property bleeding through advertising packets. *In this work, we identify such a discriminating property* and use it to associate Bluetooth randomized MAC addresses from a single device.

In short, we define our contributions as follows:

- We review literature works tracking devices and detail their limitations for our goal (section II);
- We describe our Bluetooth association strategy, highlight the main difficulty justifying the need of a weak identifier, and study it (section III);

This work has been partially funded by the ANR MITIK project, French National Research Agency (ANR), PRC AAPG2019.

[1]https://gdpr-info.eu/

- We collect three datasets of public Bluetooth advertising packets in three scenarios For users' privacy protection, the datasets were fully anonymized. Among them, a controlled environment that will serve as a ground truth (section IV);
- We analyze in detail the performance indicators we use to rate the association strategy. Our results show that the strategy is more effective in environment where user tend to spend more time, such as parks than in very dynamic environments (section IV and section V).

II. Related works

Current works on BLE tracking do not frontally attack the MAC randomization process. They either try to leverage the non-compliance of the manufacturers to the standard or exploit the vulnerabilities the Application layer introduces.

The papers [5] and [6] focus on the non-compliance. The former shows not only that manufacturers of old or low-power devices (smartwatch, wearable) either never renew the random address or advertise with the true MAC address, but also that manufacturers tend to allocate the non-random MAC addresses in batch, giving away the model of the device in the MAC prefix. The second paper shows that the sparse geographical knowledge of the user suffices to track a device, and focuses on the consequences of improper MAC randomization.

Publications [7], [8] and [9] focus on application layer-induced vulnerabilities. The papers [9] and [7] show the use Apple and Microsoft make of the `data` field in public packets–intended for general data–leaks partial identifiers: they find two *key identifiers* that are constants for a device. They exploit a mis-synchronization between the changing of the MAC and the changing of the content of the message: the device sometimes keeps the key identifiers before and after the MAC changing. Hence the name: `carry-over` algorithm. Paper [7] further analyzes the practicality of the attack; paper [8] reverse engineers the protocol Apple uses inside the public Bluetooth packets to hone the `carry-over` strategy.

In either case, the strategies give strong result and enable tracking on the long run. But if tracking non-changing MAC was reliable at the publication of [5], the situation (hopefully) changed. It appears from our dataset (see *setup B* in section IV-A) that only 4.5% to 8.4%[2] of the devices still use non-changing MAC address. We also find the carry-over algorithm only targets between 0.8% and 2.8%[2] of Apple devices. It is too restrictive to build a representative enough mobility dataset.

We propose a strategy that is less precise than the existing ones–which marks near a 100% accuracy–but targeting every device publicly advertising.

III. Bluetooth MAC association strategy

Put aside the devices not acutely following the standard and exposing the true MAC address of the device, our early experiments show that the MAC randomization event is rare.

[2]95% confidence interval on a sample of 675 addresses taken supposedly at random near a high activity center.

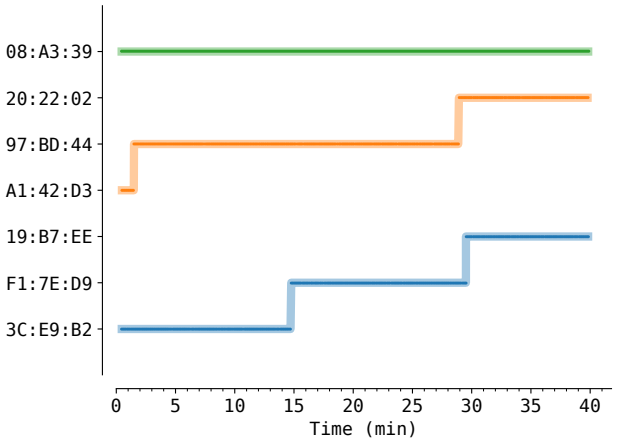

Fig. 1: MAC addresses in a controlled environment

Consider Fig.1. It shows the advertising MAC address of an iPhone 7S, an iPad Mini 1, and an iPad Pro 2019 captured in a controlled environment (as Section IV-A details) over 40 minutes. We use the same color for the different MAC addresses coming from the same device. For readability, we crop the MAC on the last three bytes.

The equivalences naturally appear as the times of disappearance and appearance visually line up. As we can see, the device using `08:A3:39` never changes its MAC; it corresponds to the iPad Mini 1, old enough not to apply BLE 4.0 properly.

The association {`A1:42:D3`, `97:BD:44`} (orange) and {`3C:E9:B2`, `F1:7E:D9`} (blue) bear no ambiguity. The difficulty arises with the associations of `F1:7E:D9` with `19:B7:EE` and `97:BD:44` with `20:22:02`: they both occur almost at the same time. These four MAC addresses are *in conflict* and require special considerations.

A. Detecting and reducing the conflicts

Conflicts are *not unlikely* because devices stop emitting during the MAC changing process. We measure four seconds for an iPhone 7S. We note d_{swap} this delay. As we will detail later, d_{swap} depends on the device.

Seeing the MACs as objects, grouping the corresponding advertising packets will prove handier: a MAC M has a beginning (M^{begin}), an end (M^{end}), and is a collection of packets.

An appearing MAC M_3, as in Fig.2, may pair with a MAC M_1 only if M_3 appears in the range $[M_1^{end}; M_1^{end} + d_{swap}]$, that is, if M_1 and M_2 validate the Boolean function:

$$C : M_i, M_j \mapsto M_i^{end} < M_j^{begin} \leq M_i^{end} + d_{swap}\,(M_i) \quad (1)$$

Two *disappearing* MAC M_1 and M_2 are in conflict if they may associate with the same MAC, *i.e.* if there exists a M_3 such that $C(M_1, M_3)$ and $C(M_2, M_3)$ are both true. With the function C, we generate all the possible conflicts (assuming we correctly calibrate d_{swap}).

In some cases, we can solve the conflicts with deduction; Fig.2 shows a typical case where carefully selecting the associations suffices to resolve the conflict, as we show hereafter. We

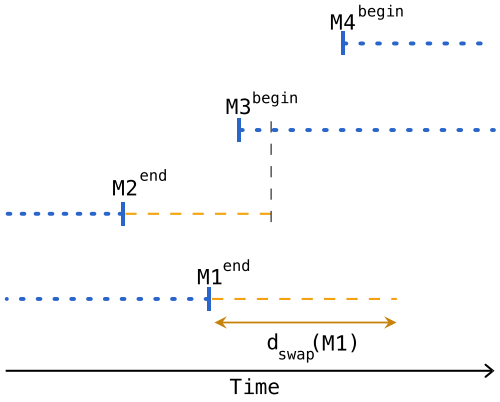

Fig. 2: Schema of two conflicting disappearing MAC. In blue, the MAC presence through time and in yellow the timespan when the new MAC should appear.

make an assumption: when a MAC disappears, we prefer the hypothesis that the sending device changed its MAC, rather than left the sensing range. This assumption is sensible for relatively static environments such as parks.

In Fig.2 M_1 and M_2 are in conflict, as they can both associate with M_3. The study of the whole set of possible associations shows that only one maximizes the number of paired devices: $\{M_1, M_3\}$ and $\{M_2, M_4\}$.

The problem boils down to solving an (unbalanced) *linear assignment problem* [10]. It aims at minimizing the cost of an association between a set of inputs (the disappearing MACs) and a set of outputs (the appearing ones). To determine the sets of disappearing and appearing MAC in conflict–the *conflict clusters*–around some MAC `M`, we compute the equivalence class of `M` in the transitive closure of C (viewed as a partial equivalence relation). For Fig.2, starting from M_1 it would generate $(\mathbf{M_1}, M_3)$, then $(M_2, \mathbf{M_3})$ and finally $(\mathbf{M_2}, M_4)$.

B. Finding a weak identifier

A *weak identifier* is a property $I \in \mathcal{I}$ of a device, deducible from its advertising packet sequences such that *(i)* $\mathcal{I}$ is big enough to resolve local conflicts (experimentally, ten properties suffice) and *(ii)* the distribution of $\mathcal{I}$ is relatively uniform across the device population. In other words, a weak identifier is a device property accessible to the sniffer, splitting the device population into a few homogeneous groups. The brand of the device or the type of advertising packet (contained in the `data` field of the packets) are (very) weak identifiers; but, they are mostly not uniform. The duration *between* advertising packets is a better option.

a) Definition of the advertising interval T_{int}: An advertising device sends an advertising packet on all three advertising channels available, within approximately 20 ms [4, p. 2754]. After such a burst, it has to wait for T_{adv} seconds, defined in eq.(2). Writing T_{var} for a "pseudo-random value [in the] range 0 ms to 10 ms" [4, p. 2751] and T_{int} for a fixed value within the range $\left[20 \text{ ms}; 10.24 \text{ s}\right]$[3],

$$T_{adv} = T_{int} + T_{var} \tag{2}$$

The device regenerates T_{var} every burst; the value T_{int} seldom changes. We call T_{int} a (device) *characteristic time*.

The characteristic time also gives insight on the capability of the device: a lower time hints at a more proactive device. We expect such devices to swap their MAC more swiftly, *i.e.* to have a lower d_{swap}. Remember that d_{swap} allows us to generate all the possible conflicts according to eq.(1). We calibrate d_{swap} as a multiple of T_{int}; $d_{swap} = 10 \times T_{int}$ gives good experimental results.

b) Measuring T_{int}: The variations of T_{var} make the measure of the characteristic time imprecise. Without further hypothesis, we can only group the times into 10 ms-wide bins. Then, the sniffer may miss packets and it listens on all three advertising channels randomly. From its point of view, assuming it missed $N-1$ packets in a row, two consecutive packets from a same device are distant by:

$$T^N_{packet} = N \times (T_{int} + T_{var}) + T_{channel}. \tag{3}$$

$T_{channel}$ is the delay the position of the channel in the advertising burst induces and T_{var} is the random variable defined earlier; we identify it with its valuation.

To compute T_{int} for a given MAC, we first get an estimate of T_{int} as $\widetilde{T}_{int}$ by picking the most filled bin. We rely on the robustness of the sniffer, so $\widetilde{T}_{int}$ differs from T_{int} by at most 10 ms (the bin width) plus $T_{channel}$. Then, we improve the estimate by taking two packets distant in time (simulating T^N_{packet} for some large N), by computing N using $\widetilde{T}_{int}$ and by getting a finer estimate of T_{int} by rewriting eq. (3) as:

$$T_{int} = \frac{T^N_{packet} - T_{channel}}{N} - T_{var} \tag{4}$$

We do not know T_{var} nor $T_{channel}$, but the bin-width (of 10 ms) absorbs them.

C. Resolving conflicts

We use T_{int} as a weak identifier; we compare them with the absolute value of the difference.

As we have also set a value for d_{swap}, we are now able to compute the conflict clusters and to resolve them. Next section details the corresponding algorithm.

D. Algorithmic resolution

The heart of the algorithm lies in the computation of conflict clusters. Let $\mathcal{P}$ be the set of all tentative associations, conflicting or not; remember eq.(1) defining C.

$$\mathcal{P} = \left\{ \left(\{M_1, \dots M_k\}, \{M'_1, \dots M'_m\} \right) | \, \forall M_i, \exists M'_j, \; C(M_i, M'_j) \right\} \tag{5}$$

$\mathcal{P}$ is a set of pairs breaking a conflict cluster into the set of disappearing MACs and the set of appearing MACs; A non-ambiguous association of M_1 and M_2,such as in Fig.1, would have the form: $(\{M_1\}, \{M_2\})$.

[3]The lower bound depends on the kind of advertising packet the device sends. It does not matter here.

To resolve the conflicts, we use a distance comparing the weak identifiers; we write it D_{id}, such that:

$$D_{id}(M_i, M_j) = \begin{cases} d_{swap}(M_i) - d_{swap}(M_j) & \text{if } C(M_i, M_j) \\ \infty & \text{otherwise} \end{cases} \quad (6)$$

It is the difference between the characteristic times, or an infinite value when the association is impossible.

With these notations, we write the algorithm 1.

Algorithm 1 Association strategy algorithm

Input: The set of advertising packets $\mathcal{M}$

Compute $\mathcal{T}$ the set of characteristic time
Compute $\mathcal{P}$ using $\mathcal{T}$
$Association \leftarrow \{\}$
for all (S, S') in P **do**
  $LocalAssoc \leftarrow$ `LinearAssignment`(S, S', D_{id})
  $Association \mathrel{+}= LocalAssoc$
end for

The `Union Find` algorithm helps compute $\mathcal{P}$ in a reasonable time [11]. The Hungarian algorithm [12] solves `LinearAssignment` and runs in strongly quadratic time with respect to the number of MAC in conflict at the same time. We see in section IV-B it does not impede the performances.

IV. Performances

This section presents the validation of our association algorithm. The meaning of *validation* is fourfold here:

(1) Ensuring the algorithm will run in a reasonable amount of time;
(2) Confirming the functioning of the principle;
(3) Attesting the quality of the characteristic time as a weak identifier;
(4) Verifying the algorithm pairs the different MAC of a device.

To ensure the algorithm will run in a reasonable amount of time (1), we have to show the conflict clusters are generally small. To confirm the algorithm works in principle (2), it suffices to test it on a small, controlled environment. To verify the characteristic time is a weak identifier (3), we have to see if it breaks devices into sufficiently many homogeneous groups. But to verify the algorithm correctly associate many MAC in a busy environment (4), we cannot rely on a ground truth other than what the state-of-the-art tracking algorithms may generate. The only applicable here is the `carry-over` algorithm [9] and we show it is ineffective on our datasets. We can but verify, either with quantitative discriminating packet content, or with qualitative environmental observations, that the associations are coherent. Yet, any discriminating packet content being a weak identifier, we could improve *our* weak identifier by combining it with the packet content. This new weak identifier would be more discriminative–*i.e.* break the set of devices into more classes–and the algorithm would be more effective. So we are bound to verify our algorithm with sub-optimal parameters.

Nonetheless, the following sections tackle each of these points. Section IV-A details the hardware and the datasets, section IV-B shows the practical execution time is fair and section V analyses the functioning of the algorithm and different performance markers.

Model	iPhone SE	iPad 6	iPad 7	iPad Pro 3	iPad Mini 2	iPhone 7
iOS	13.3.1	13.4.1	13.4	13.3.1	12.4.6	13.4.1

TABLE I. Hardware and OS version we use for the *Controlled Environment* dataset.

A. Experimental setup

Advertising packets are public, their capture does not require specialized hardware. We use the internal Bluetooth chipset of a battery-powered Raspberry Pi 4 and the free drivers `hcitool` and `btmon` [13]. Using this hardware, we build three datasets to test our algorithm: one in a controlled environment, two in uncontrolled environments. Each of them, described below, tests a specific usage pattern. It is important to notice that, to protect users privacy, we fully anonymize the data we collect by consistently replacing the (randomized) MAC addresses with a 48 bits random value. We also, do not store any information that could identify individual users. Finally, as [9] points out, Android phones unlikely advertise themselves. The datasets will be mostly biased toward Apple devices.

a) Controlled Environment: We capture the packets coming *only* from the 6 devices of table I over 40 minutes. We *control* the devices; we can deduce a ground truth using optional packet information.

b) Uncontrolled environment: We capture adverting packets for about an hour in two environments: one, noted *setup A*, extremely heterogeneous with more than a thousand devices, and the other, *setup B*, where about 50 devices remain in the sensing range for the whole measure. In both *setup A* and *setup B*, we witness two mobility classes: the very dynamics, staying less than 60 seconds in the sensing range and accounting for the vast majority of devices, and the very static, staying in range during the whole measure. *Setup A* will be a stress test; *setup B* will be a validation test.

To highlight mobility classes, we plot the cumulative distribution function (CDF) of the MAC lifespan in Fig. 3. The steep left-hand side of the curve corresponds to the dynamic class: most devices spend less than two minutes in sensing range.

MAC randomization breaks the "natural" shape of the curve, especially in *setup A*, introducing a step. MAC randomization, until BLE 4.1, forces devices to change their MAC precisely every 15 minutes; hence the disproportion of MAC living for 15 minutes. *Setup B* does not shows such a step probably because of its size.

B. Estimation of the conflict cluster size

The bottleneck of the Algorithm 1 is the computation of all the `LinearAssignment` instances, which is strongly quadratic in the size of the conflict cluster. To show the algorithm will run in a reasonable time, we plot the cumulated distribution of the sizes of cluster conflicts in *setup A* in Fig.4. As the dataset

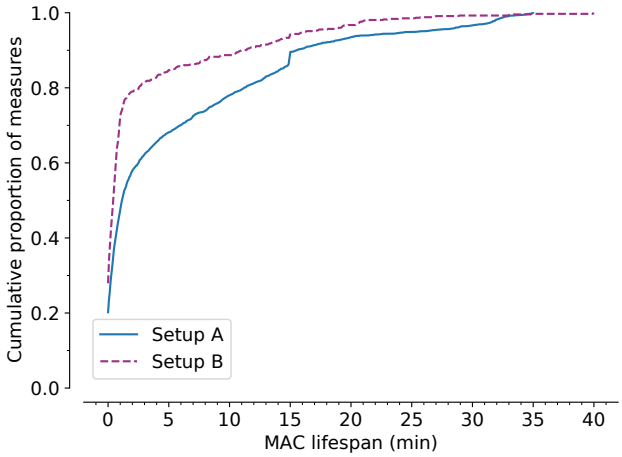

Fig. 3: Cumulated distribution of the MAC lifespan on in *setup A* and *B*.

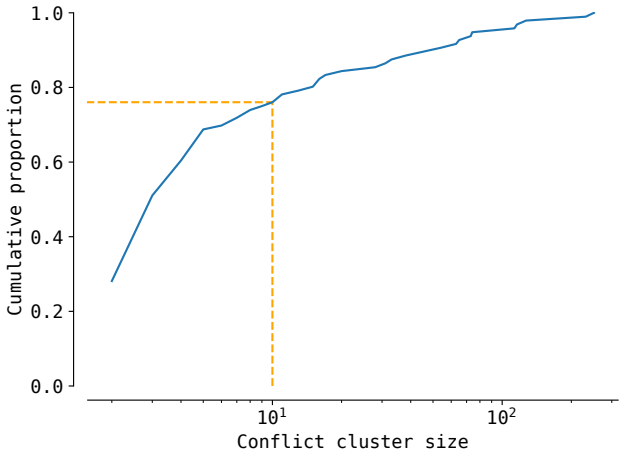

Fig. 4: Cumulated distribution of the conflict cluster size in *setup A*.

is a stress test, it represents an expected worst-case scenario. 78% of the clusters involve fewer than ten devices. Hence, the algorithm will not suffer from the quadratic complexity. In practice, our algorithm handles *setup A* containing about 2500 MACs in less than 10 seconds on an `Intel i7,` 10^{th} `Gen`.

The few clusters gathering a hundred devices or more are a weakness of the algorithm: the construction of $\mathcal{P}$ gives equal importance to all the associations (before the weighting) to ease its computation. Stair-like sequences of appearing and disappearing MAC will expand a conflict cluster over time; MAC with short lifespan amplify the process.

V. Performance markers

Fig.5 depicts the automatic associations using our proposed algorithm on the controlled dataset, which is small enough to

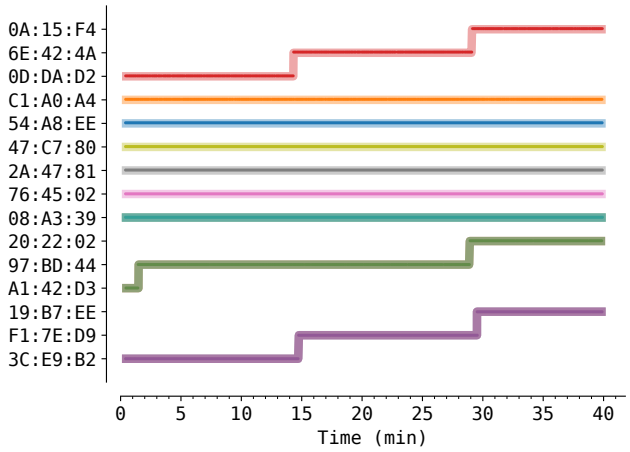

Fig. 5: MAC associations for the controlled environment. Same colors for a same device.

deduce the ground truth. As we can see, we associate, with no errors, MAC addresses that disappear at the same time.

For the other two datasets, if the algorithm works properly, it should: (a) Keep the associations the `carry-over` [9] strategy proposes; (b) Make the step at 15 minutes in figure 6 disappear; (c) Find about 50 static devices in *setup B*; and finally (d) Consider the MACs with different optional packet information as coming from different devices. In the following, we tackle all these propositions.

a) `Carry-over` *comparative results:* As described in section II the `carry-over` algorithm relies on two key elements in the packets. In either dataset, fewer than 30 devices had both, and the algorithm does not manage any association. Trying to associate a device using only one key gave inconsistent results. Apple appears to have patched the vulnerability.

b) Reducing the 15-minute peak: As noted above, many devices of the *setup A* dataset use the recommended Bluetooth parameters and change their MAC address every 15 minutes. It translates to a step at 15 minutes in Fig.3 for the CDF. Thus, to test the effectiveness of our association algorithm we can verify it detects all the 15-minutes-long MAC address as changing MAC. More precisely, the peak at 15 minutes should disappear, and a step for the devices that remained the whole capture should appear at the end of the measure. In Fig.6, we plot the CDF of the captured MAC lifespan and the CDF of the compound MAC lifespan. The compound MAC corresponds to the lifespan of a device after the association of multiple MAC addresses. We remove from the original CDF the MACs that our algorithm groups together as the same device, and we add the compound duration.

As we can see, in Fig.6, the peak entirely disappears, and a step appears at the end, although not as sharp as a perfect algorithm would produce. There is room for improvement.

c) Real population in setup B: There were about 50 people in range during the capture. We remove the passers-by (MAC lifespan inferior to 60 seconds); they would wrong the

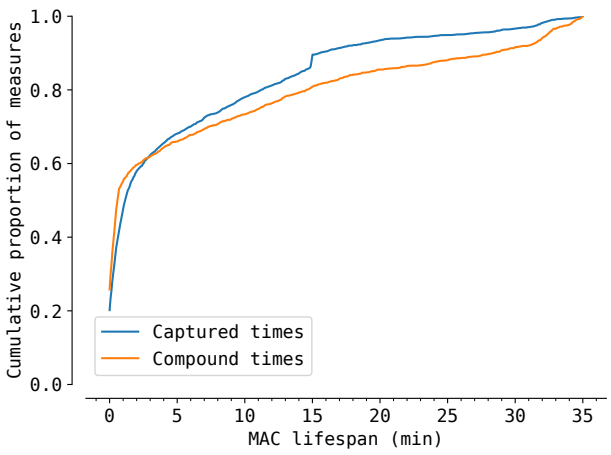

Fig. 6: Cumulated distribution function of the lifespan of the different aggregated and captured MAC of *Setup A*.

results because the algorithm assumes people tend to stay in the capture, and because all the passer-by would count as an individual.

When executing our association algorithm, we find 43 different devices, which is coherent with the number of expected devices. All people may not have had a phone. The fact that the number of found devices is close to the expected number of devices allows us to assume that our strategy works.

d) Optional parameter mis-associations: The standard defines four kinds of advertising packets, and devices mostly use two of them. A device will not use two different types with the same MAC. Even when changing MAC, a device will generally keep the same kinds of advertising messages. This is (almost) a device property, and our associations should not break it. The same goes for the brand of the device, also specified in the packet.

We measure, for both datasets, the number of times the algorithm wrongly associates the two advertising types. This measure alone would sometimes count the same error twice, so we add the number of compound MAC with at least one parameter mis-association. Table II shows the results in the form `Number of mismatches / Number of pairs or components`

In *setup A*, the algorithm makes less than 5% of mismatches (both on the brand and the advertising type), and nears 10% errors on the component-wise measure for advertising types in *setup B*. The results show that most of the errors happen once in the component and that other optional packet parameters have little discrimination power. Other weak identifiers probably exist in the data field of the packets or at the Link layer.

VI. Conclusion

In this paper, we proposed a general algorithm to take the low frequency of MAC changing at our advantage to associate different MACs from the same device. For this, we rely on a weak identifier–the timing between advertising packets–and weigh the tentative associations with the distance between their weak identifiers. We tested our strategy on a small but controlled dataset; as the algorithm benefits from a low number of devices, we applied it on two more significant stress tests. In these cases, as it is close to impossible to get a ground truth, we used markers hinting at the method success. Should researchers find a better weak identifier, the algorithm would provide better results.

Dataset	**Pair-wise mismatch**		**Component-wise mismatch**	
Association parameter	Brand	Adv. type	Brand	Adv. type
Setup A	5/1087	24/1088	4/388	17/388
Setup B	0/66	3/66	0/31	3/31

TABLE II. Mismatch results for the Advertising packets and the devices brand.

The next step for this work is a theoretical analysis, both on the probability of conflicts to deduce formal requirements on the notion of weak identifier and the weak identifier itself, *i.e.* on the probability of discrimination *in practice*, two devices.

References

[1] "IEEE standard for information technology-telecommunications and information exchange...," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, vol. 11, pp. 1–3534,

[2] B. SIG, *Specification of the Bluetooth System, Core v4.1*. Mar. 12, 2013.

[3] ——, *Specification of the Bluetooth System, Core v4.0*. Jun. 30, 2013.

[4] ——, *Specification of the Bluetooth System, Core v5.1*. Jan. 21, 2019.

[5] J. Martin, T. Mayberry, C. Donahue, *et al.*, "A study of mac address randomization in mobile devices and when it fails," *PoPETs*, vol. 2017, no. 4, pp. 365–383, 2017.

[6] T. Issoufaly and P. U. Tournoux, "Bleb: Bluetooth low energy botnet for large scale individual tracking," in *NextComp*, IEEE, 2017, pp. 115–120.

[7] G. Celosia and M. Cunche, "Saving private addresses: An analysis of privacy issues in the bluetooth-low-energy advertising mechanism," in *MOBIQUITOUS*, 2019, pp. 444–453.

[8] J. Martin, D. Alpuche, K. Bodeman, *et al.*, "Handoff all your privacy–a review of apple's bluetooth low energy continuity protocol," *PoPETs*, vol. 2019, no. 4, pp. 34–53, 2019.

[9] J. K. Becker, D. Li, and D. Starobinski, "Tracking anonymized bluetooth devices," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 50–65, 2019.

[10] S. Martello and P. Toth, "Linear assignment problems," in *North-Holland Mathematics Studies*, vol. 132, Elsevier, 1987, pp. 259–282.

[11] G. C. Harfst and E. M. Reingold, "A potential-based amortized analysis of the union-find data structure," *ACM SIGACT News*, vol. 31, no. 3, pp. 86–95, 2000.

[12] J. Munkres, "Algorithms for the assignment and transportation problems," *SIAM*, vol. 5, no. 1, pp. 32–38, 1957.

[13] Bluez Home. [Online]. Available: https://www.bluez.org.